**Міністерство освіти і науки України**

**Чернівецький національний університет імені Юрія Федьковича**

**Інститут фізико-технічних та комп’ютерних наук**

(повна назва інституту/факультету)

**Кафедра комп’ютерних наук**

(повна назва кафедри)

КУРСОВА РОБОТА

з дисципліни

«Інтелектуальний аналіз засобами Python/R»

на тему: «**РЕКОМЕНДАЦІЙНА СИСТЕМА ТУРИСТИЧНОЇ СОЦМЕРЕЖІ НА ОСНОВІ АЛГОРИТМУ МАТРИЧНОЇ ФАКТОРИЗАЦІЇ**»

Виконав: студент 4 курсу, групи 444

Напряму підготовки 122 «Комп’ютерні науки»

(шифр і назва спеціальності)

Скібінський Денис Русланович .

(прізвище та ініціали)

Керівник \_к.б.н., асистент Талах М.В.\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Національна шкала\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_

Кількість балів \_\_\_ \_ Оцінка: ECTS\_ \_\_

Чернівці – 2021

**АНОТАЦІЯ**

У курсовій роботі розроблено додаток, в якому було реалізовано рекомендаційну систему для туристичної соцмережі на основі методу матричної факторизації. Були досліджені види рекомендаційних систем та описано методи колаборативної фільтрації. Детально описаний алгоритм матричної факторизації та його реалізація в додатку. Показана працездатність цього алгоритму в створеному додатку.

Пояснювальна записка складається з 3 розділів загальним обсягом 22 сторінки, містить 2 додатки та 6 використаних джерел.

**Ключові слова:** ML.NET, C#, рекомендаційна система, колаборативна фільтрація, матрична факторизація.

**ЗМІСТ**

[**ВСТУП** 4](#_Toc91104956)

[**РОЗДІЛ 1. РЕКОМЕНДАЦІЙНІ СИСТЕМИ, ЇХ ВИДИ ТА СФЕРИ ЗАСТОСУВАННЯ** 6](#_Toc91104957)

[**1.1 Означення рекомендаційної системи.** 6](#_Toc91104958)

[**1.2 Види рекомендаційних систем.** 6](#_Toc91104959)

[**1.3 Обґрунтування вибору алгоритму для вирішення поставленої задачі.** 7](#_Toc91104960)

[**1.4 Колаборативна фільтрація.** 8](#_Toc91104961)

[**1.5 Проблеми колаборативної фільтрації.** 8](#_Toc91104962)

[**1.6 Огляд існуючих додатків з рекомендаційними системами.** 9](#_Toc91104963)

[**РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ДОДАТКУ** 10](#_Toc91104964)

[**2.1 Проектування та вибір технологій.** 10](#_Toc91104965)

[**2.2 Матрична факторизація.** 11](#_Toc91104966)

[**2.3 Метод стохастичного градієнта** 12](#_Toc91104967)

[**2.4 Отримання та обробка вхідних даних.** 12](#_Toc91104968)

[**2.5 Створення та навчання моделі.** 13](#_Toc91104969)

[**2.6 Використання створеної моделі.** 14](#_Toc91104970)

[**РОЗДІЛ 3. ДЕМОНСТРАЦІЯ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ** 15](#_Toc91104971)

[**ВИСНОВКИ** 18](#_Toc91104972)

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** 19](#_Toc91104973)

[**ДОДАТКИ** 21](#_Toc91104974)

**ВСТУП**

Нині ми живемо в час інформації. З розвитком інтернету кількість інформації, з якою ми стикаємось кожен день, зросла до неймовірних чисел. З року в рік всесвітня мережа поповнюється величезною кількістю контенту різного роду і жанру. Фотографії в Instagram, відео в YouTube, пісні в Spotify – інформаційна база збільшується в геометричній прогресії. Постає питання: як розібратись в цьому морі інформації і знайти те, що необхідно саме вам? Вирішення цієї проблеми полягає в персоналізації інформації, отриманні такого контенту, який буде цікавий та потрібний вам. Вирішення цієї задачі є завданням рекомендаційної системи. Така система могла би пришвидшити пошук потрібних даних, зекономити час та полегшити роботу користувача з додатком.

Об’єктом дослідження курсової роботи є рекомендаційна система. Предметом дослідження курсової роботи є дослідження методу колаборативної фільтрації на основі алгоритму матричної факторизації та його застосування для рекомендації подорожей користувачу туристичної соцмережі.

Метою написання курсової роботи є створення програмного додатку, який буде реалізовувати рекомендаційну систему туристичної соцмережі. До завдань курсової роботи належать наступні завдання: аналіз методів для створення рекомендаційних систем, вибір алгоритму для реалізації конкретної рекомендаційної системи, отримання та підготовка даних для подальшого використання в алгоритмі навчання, створення та навчання моделі на основі обраного алгоритму, перевірка створеної моделі, створення інтерфейсу для використання моделі. Розроблений рекомендаційний модуль можна пізніше інтегрувати в готовий застосунок туристичної соцмережі для аналізу та персоналізації даних.

В першому розділі розглянуто основні види рекомендаційних систем, описано метод колаборативної фільтрації та обґрунтовано його використання для досягнення поставленої, проаналізовано існуючі популярні додатки, які використовують рекомендаційні системи на основі колаборативної фільтрації.

В другому розділі описано проектування додатку, використані технології, вхідні та вихідні дані модулю та конкретну реалізацію методу колаборативної фільтрації.

В третьому розділі описано та продемонстровано практичне використання створеного програмного продукту.

**РОЗДІЛ 1. РЕКОМЕНДАЦІЙНІ СИСТЕМИ, ЇХ ВИДИ ТА СФЕРИ ЗАСТОСУВАННЯ**

**1.1 Означення рекомендаційної системи.**

Рекомендаційна система – це підклас систем фільтрації інформації, створений для прогнозування оцінки чи вподобання користувачем певних даних.

Рекомендаційні системи активно застосовуються в різних галузях. Прикладами використання є різноманітні генератори плейлістів для музичних та відеосервісів, рекомендаційні системи товарів в інтернет-магазинах, контенту в соціальних мережах тощо. Ці системи обробляють вхідні дані з одного ресурсу (музика) або декількох платформ (новини, книжки, пошукові запити). [[1](#СПИСОК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)]

**1.2 Види рекомендаційних систем.**

Більша частина рекомендаційних систем використовує два класи інформації:

* *Інформація про взаємодію користувача з об’єктом* – історія переглядів, рейтинги, оцінки користувача, списки улюблених елементів;
* *Інформація про об’єкт* – характеристики об’єкту (назва, опис, категорія).

Базуючись на цих класах інформації розрізняють наступні види фільтрацій для рекомендаційних систем:

* *Колаборативна фільтрація* – модель будується на основі минулої поведінки користувача (придбані чи переглянуті товари, оцінка користувача цих товарів), а також схожих рішень, зроблених іншими користувачами. Ця інформація використовується для прогнозування рейтингу та формування списку рекомендацій інших товарів;
* *Фільтрація на основі вмісту* – використовує набір дискретних характеристик товару для того, щоб рекомендувати користувачу товари зі схожими властивостями;
* *Гібридна фільтрація* – використовує переваги обох вищезгаданих методів шляхом окремого їх використання, або інтеграцій одного з методів в інший. Гібридна фільтрація також допомагає подолати певні проблеми, які виникають при поодинокому використанню цих методів.

Дані, які використовуються рекомендаційними системами, можуть збиратись явними та неявними методами, та їх поєднаннями.

Приклади явного збору даних:

* Користувачу оцінює об’єкт за диференційованою шкалою;
* Користувач ранжує групу об’єктів від найкращого до найгіршого;
* Користувач вибирає кращий з двох об’єктів;
* Користувач створює список улюблених об’єктів.

Приклади неявного збору даних:

* Спостереження за переглянутими користувачем об’єктами;
* Ведення журналу дій користувача.

**1.3 Обґрунтування вибору алгоритму для вирішення поставленої задачі.**

Для вирішення задачі створення рекомендаційної системи туристичної соцмережі було вирішено використати метод колаборативної фільтрації з ряду причин.

Однією з цих причин є те, що використання цього методу не потребує інформації про сам об’єкт фільтрації, а лише про взаємодію користувачів з об’єктом. В туристичній мережі, на основі даних якої будується модель, вже існує створена і функціонуюча система рейтингів, а отже і існують дані для аналізу і навчання моделі.

Другою причиною є те, що рекомендації в колаборативній фільтрації здатні розширювати інтереси користувачів. На відміну від фільтрації на основі вмісту, колаборативна фільтрація не обмежена характеристиками об’єкту, а формує рекомендації на основі схожих думок користувачів, які можуть бути прив’язані не тільки до якісних характеристик цього об’єкту.

**1.4 Колаборативна фільтрація.**

Колаборативна фільтрація – одна з методів, який використовується рекомендаційними системами.

Ідея колаборативної фільтрації полягає в створенні прогнозів (фільтрації) стосовно інтересів користувача на основі вподобань та смаків інших користувачів (колаборації). Узагальнено ідея звучить наступним чином: якщо користувач А має ту саму думку стосовно об’єкту, що і користувач Б, то думки А та Б будуть схожими стосовно й інших об’єктів. [[2](#СПИСОК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)]

Узагальнений алгоритм системи колаборативної фільтрації виглядає наступним чином:

1. Користувач висловлює свою думку стосовно об’єкту (книжки, фільму, пісні) у вигляді певної оцінки.
2. Система порівнює оцінку користувача з оцінками інших користувачів та знаходить людей з найбільш схожими смаками.
3. Система рекомендує користувачу об’єкти, які сподобалися “схожим” користувачам, але ще не оцінені поточним користувачем.

**1.5 Проблеми колаборативної фільтрації.**

Колаборативна фільтрація стикається з рядом проблем та недоліків:

* *Розрідженість даних* – більшість користувачів не ставить оцінки товарам, тому матриця «предмет-користувач» виходить розрідженою, що ускладнює обчислення рекомендацій, що особливо відчутно в щойно створених системах;
* *Масштабованість* – зі збільшенням кількості користувачів алгоритм колаборативної фільтрації стає занадто складним для розрахунків;
* *Проблема холодного старту* – нові об’єкти та користувачі створюють проблему рекомендаційним системам;
* *Шахрайство* – люди можуть використовувати оцінки для реклами своїх товарів та усунення товарів конкурентів;
* *Білі ворони* – користувачі, чия думка не збігається з думкою більшості, що ускладнює рекомендацію. [[3](#СПИСОК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)]

**1.6 Огляд існуючих додатків з рекомендаційними системами.**

Колаборативна фільтрація ефективно використовується в рекомендаційних системах ряду інтернет-сервісів та додатків. Рекомендації стали звичною для нас справою при їх використанні і ми навіть не задумуємось наскільки ми залежні від них. Нижче наведені приклади сервісів з рекомендаційними системами на основі методу колаборативної фільтрації:

**Reddit** – розважальний, новинний онлайн-сервіс, а також інтернет ЗМІ, де зареєстровані користувачі можуть додавати свій контент, такий як текстові пости або прямі посилання й обговорювати їх. Як і багато інших подібних сайтів, Reddit підтримує систему голосування за вподобані повідомлення — найбільш популярні з них з'являються на головній сторінці сайту.

**Instagram** – соціальна мережа, що базується на обміні фотографіями, дозволяє користувачам робити фотографії, застосовувати до них фільтри, а також поширювати їх через свій сервіс і низку інших соціальних мереж.

**YouTube** – популярний відеохостинг, що надає послуги розміщення відеоматеріалів. Є підрозділом компанії Google. користувачі можуть додавати, переглядати і коментувати ті чи інші відеозаписи. Завдяки простоті та зручності використання, YouTube став одним із найпопулярніших місць для розміщення відеофайлів. Служба містить як професійні, так і аматорські відеозаписи, у тому числі відеоблоги.

**РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ДОДАТКУ**

**2.1 Проектування та вибір технологій.**

Програмний рекомендаційний модуль для туристичної соцмережі повинен мати зручну структуру для подальшого використання та бути легким в модифікації.

Оскільки додаток туристичної соцмережі написаний на платформі .NET та мові програмування C#, то для реалізації рекомендаційної системи було вирішено також використати .NET та бібліотеку машинного навчання ML.NET, що могло би полегшити підтримку та масштабування модулю.

Бібліотека ML.NET реалізовує алгоритм матричної факторизації на основі стохастичного градієнтного спуску для створення рекомендаційних систем. Саме його і буде використано в програмному додатку.

Сам додаток буде створений у вигляді Web API, має працювати з базою даних додатку та надавати список рекомендованих подорожей для даного користувача. Для поставлених задач було обрано наступні технології:

* мова програмування **C#**;
* **ASP .NET** – платформа для створення веб-додатків на мові C#, використана для створення Web API;
* **ML .NET** – бібліотека машинного навчання для мови С#;
* **AutoMapper** – бібліотека для мапінгу об’єктів всередині програми;
* **Entity Framework Core** – бібліотека для доступу даних додатку з бази даних;
* **MS SQL Server** – реляційна система керування базами даних, використана для збереження даних;
* **Swagger** – бібліотека для створення графічного інтерфейсу API

**2.2 Матрична факторизація.**

Матрична факторизація – це клас алгоритмів колаборативної фільтрації, який використовується в рекомендаційних системах. Ідея алгоритму полягає в декомпозиції матриці взаємодії користувача та об’єкту в добуток двох прямокутних матриць меншої розмірності.

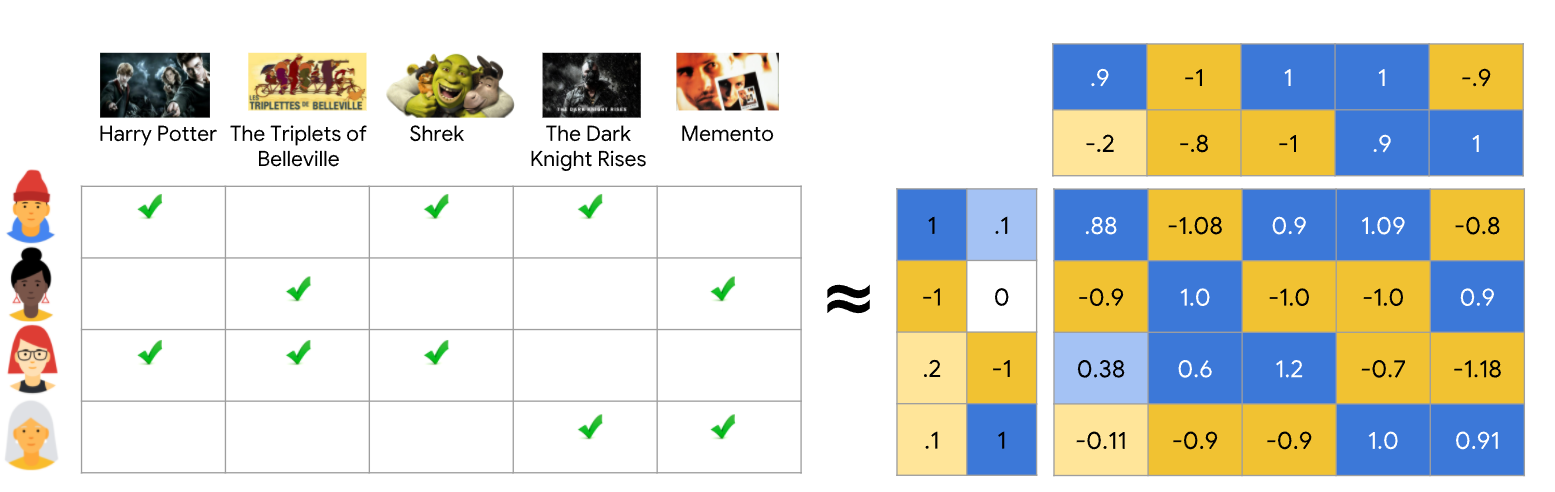


Рисунок 2.1 - Матриця взаємодії користувача та об’єкту

Для кращого розуміння використання матричної факторизації, розглянемо приклад рекомендації фільмів. Нехай ідентифікатори користувачів та фільмів – значення індексів рядків та стовпців матриці, а значення – рейтинги фільмів, надані цими користувачами. Тобто значення *r* в рядку *u* та стовпці *v* будеозначати, що користувач *u* дав оцінку *r* фільму *v*. Незавершена матриця – звичне явище, тому що не всі користувачі надають оцінку усім об’єктам. Допустимо, що R∈Rm×n – це матриця розміром *m* на *n* а ранг двох факторних матриць – P∈Rk×m та Q∈Rk×n, де *k* – ранг наближення. Спрогнозованою оцінкою в рядку *u* та стовпці *v* матриці *R* буде внутрішній добуток рядка *u* матриці *P* та стовпця *v* матриці *Q*, що означає, що *R* є наближення до добутку транспонованої матриці *P* на матрицю *Q*. Ранг *k* зазвичай набагато менший за *m* та *n*, тому цей добуток часто називають низькоранговим наближенням *R*. [[4](#СПИСОК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)]

**2.3 Метод стохастичного градієнта**

Для пошуку матриць P та Q застосовується метод стохастичного градієнту. Стохастичний градієнтний спуск – ітеративний метод оптимізації градієнтного спуску за допомогою стохастичного наближення. Використовується для прискорення пошуку цільової функції шляхом використання обмеженого за розміром тренувального набору, який вибирається випадкового при кожній ітерації. [[5](#СПИСОК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)]

**2.4 Отримання та обробка вхідних даних.**

Дані для аналізу та обробки знаходяться в реляційній базі даних додатку. Нижче наведена схема бази даних додатку.

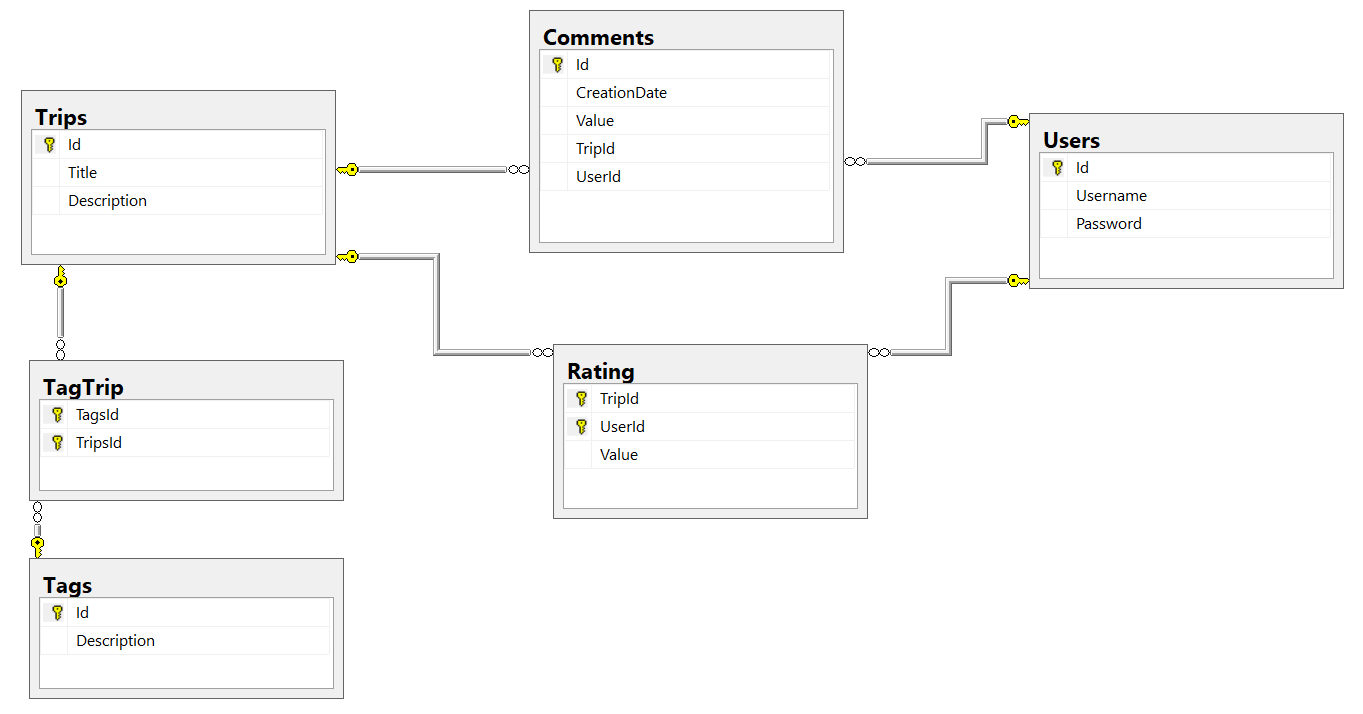


Рисунок 2.2 – Схема даних додатку

Для формування списку рекомендованих подорожей користувача ми повинні мати інформацію про вподобання користувачів. В додатку реалізована система рейтингів – кожен користувач може оцінити подорож цілим числом від 1 до 5. Інформація про рейтинги знаходиться в таблиці Rating. Вона складається з трьох полів: ідентифікатор подорожі, ідентифікатор користувача та значення оцінки. Саме з цих даних ми сформуємо матрицю, яку використаємо для навчання моделі.

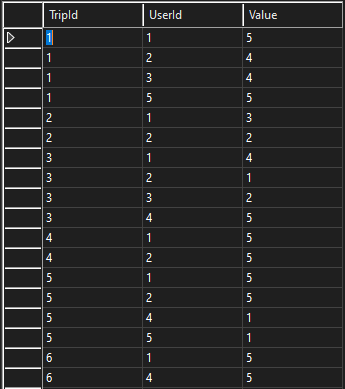


Рисунок 2.3 – Дані в таблиці Rating

Оскільки дані валідуються і зберігаються в зручному форматі, то вони не потребують попередньої обробки.

**2.5 Створення та навчання моделі.**

Весь процес створення та навчання моделі був інкапсульований в клас з назвою LearningService, який має єдиний публічний метод Learn, я якому зосереджена вся логіка створення та навчання моделі. Код сервісу наведений у додатку А.

Процес навчання можна звести до декількох етапів:

1. Створення об’єкту класу MLContext, який є вхідною точкою для роботи з нашою моделлю
2. Створення тренувальної вибірки типу IDataView на основі інформації, отриманих з бази даних.
3. Створення пайплайну обробки даних в моделі та вибір алгоритму (в нашому випадку це алгоритм матричної факторизації).
4. Тренування моделі (метод Fit)
5. Збереження натренованої моделі в файл model.zip

В результаті навчання ми отримаємо готову модель, яку можемо використовувати для прогнозування. [[6](#СПИСОК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)]

**2.6 Використання створеної моделі.**

Навчена модель збережена у файлі model.zip у папці додатку. Бібліотека ML .NET надає метод розширення для платформи ASP .NET, який дозволяє використовувати механізм Dependency Injection для автоматичного створення та оновлення класу PredictionEngine на основі збереженої у файлі моделі. Цей клас ми й будемо використовувати для формування прогнозу та рекомендованих користувачу подорожей.

Логіка прогнозів була винесена в окремий сервіс PredictingService, код якого знаходиться в додатку Б.

Процес формування рекомендаційного списку подорожей відбувається наступним чином:

1. В метод GetRecommendedForUser передається ідентифікатор користувача.
2. З бази даних отримується список ідентифікаторів подорожей, ще не оцінених даним користувачем.
3. Для кожної пари користувач-подорож прогнозується рейтинг на основі раніше навченої моделі.
4. Метод повертає набір подорожей разом з спрогнозованими рейтингами для даного користувача.

Отримані дані можна пізніше відфільтрувати в залежності від бажаного рейтингу. Зокрема можна повернути лише ті подорожі, для яких спрогнозований рейтинг більше якогось конкретного числа.

**РОЗДІЛ 3. ДЕМОНСТРАЦІЯ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ**

Запущений додаток API надає для використання декілька HTTP запитів, які можна викликати через графічний інтерфейс Swagger, чи через сторонні додатки, які можуть надіслати HTTP запити. Саме таким чином буде відбуватися комунікація додатку туристичної соцмережі з цим модулем. Нижче наведений графічний інтерфейс Swagger.

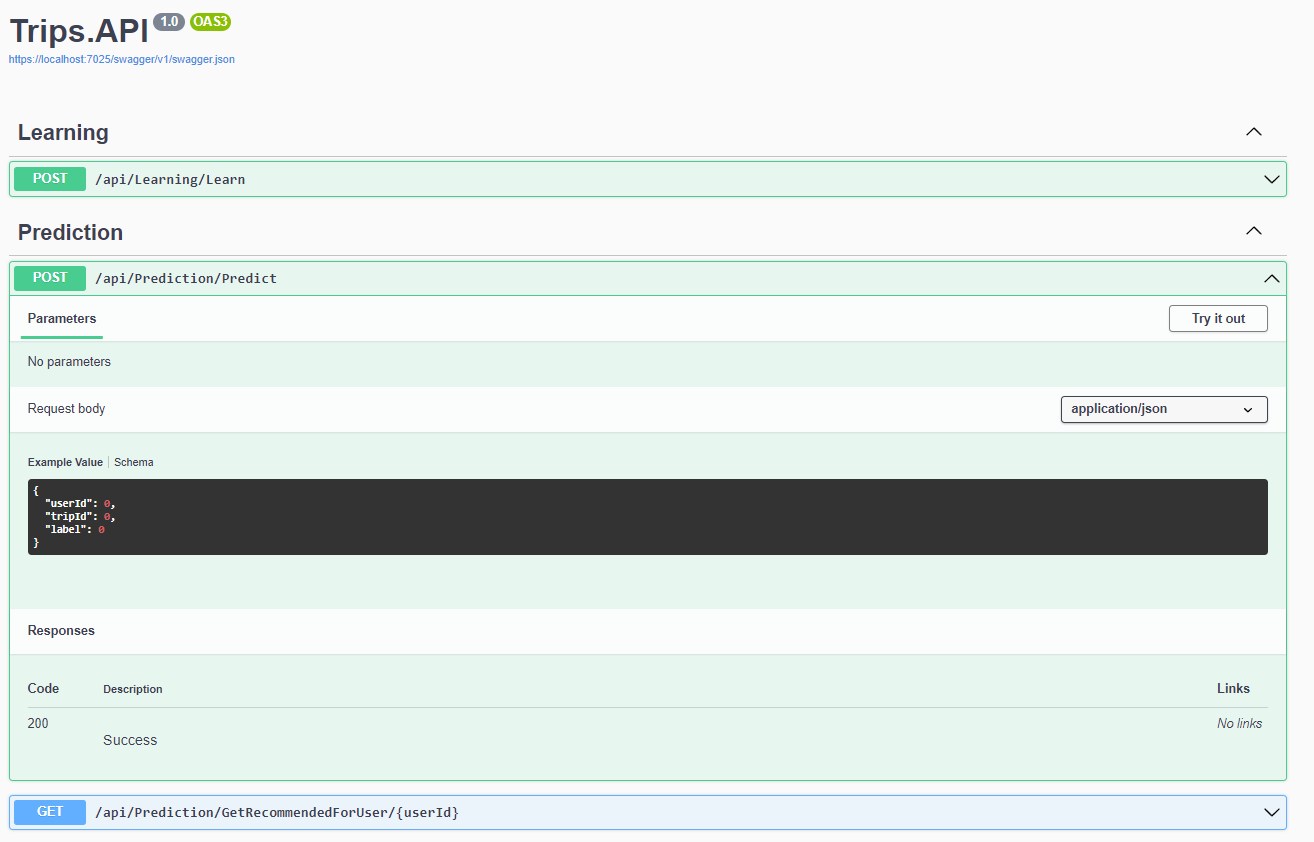


Рисунок 3.1 – Графічний інтерфейс API

Графічний інтерфейс дозволяє протестувати роботу API шляхом введення необхідних параметрів запиту та подальшої відправки запиту.

Кожен метод API наділений механізмом обробки помилок: у разі виникнення помилки, запит поверне статусний код та інформацію про помилку. Додаток, що використовує API, може здійснити обробку помилок в залежності від повернутого статусного коду та вивести повідомлення помилки.

Для навчання моделі на основі інформації з бази даних необхідно викликати POST-запит */api/Learning/Learn*. Він отримає дані про рейтинг подорожей, створить та навчить рекомендаційну модель, а потім збереже її у файл model.zip в кореневій папці програми задля подальшого її використання при прогнозах.

Навчену модель можна використати для прогнозу рейтингу конкретним користувачем деякої подорожі. Для цього необхідно використати POST-запит */api/Prediction/Predict*. В тілі запиту необхідно вказати ідентифікатори користувача та подорожі, для яких ми хочемо отримати прогноз. На рисунку нижче зображено вигляд тіла запиту. Результатом запиту буде об’єкт з інформацією про прогнозовану оцінку користувачем подорожі, яка являє собою десятковий дріб з проміжку від 1 до 5.

Основною функцією додатку є створення рекомендаційного списку. Для його отримання необхідно виконати GET-запит */api/Prediction/GetRecommendedForUser/{userId}*. Єдиним параметром цього запиту буде ідентифікатор користувача, для кого ми формуємо список рекомендованих подорожей. Нижче наведено приклад рекомендаційного списку для користувача з ідентифікатором 4.

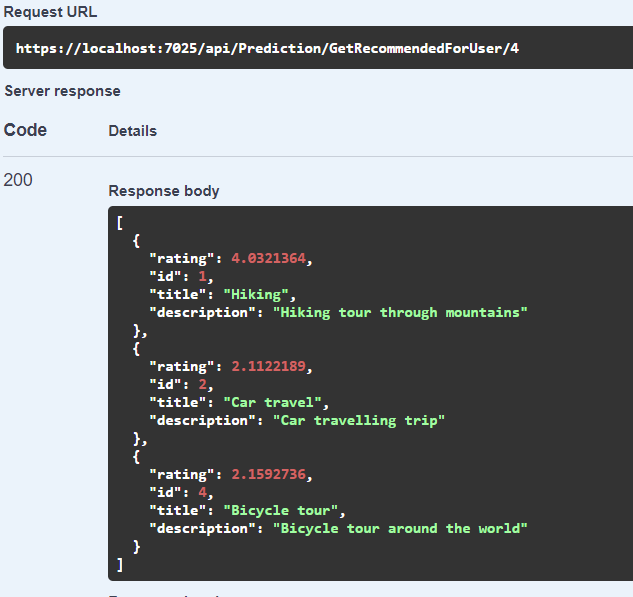


Рисунок 3.2 – Приклад рекомендаційного списку для користувача

API оперує даними в JSON форматі, що є зручним для обробки веб-додатками, оскільки більшість з них оперує бібліотеками серіалізації та десеріалізації JSON об’єктів.

**ВИСНОВКИ**

Для виконання поставленої задачі було проведено аналіз існуючих підходів та методів для створення рекомендаційних систем, оглянуто вже існуючі рекомендаційні системи популярних сервісів та обрано конкретний метод для його подальшої реалізації в додатку.

В результаті виконання курсової роботи було розроблено API, яке аналізує збережені додатком дані та формує модель для прогнозування персоналізованого рейтингу подорожей для користувача на основі методу колаборативної фільтрації та алгоритму матричної факторизації. Створений додаток містить зрозумілий графічний користувацький інтерфейс, що дозволяє використовувати його як самостійний елемент програмного забезпечення, а архітектура додатку у вигляді API дозволяє легко інтегрувати його як модуль для іншого програмного продукту.

Внаслідок виконання курсової роботи було успішно досягнуто мети й виконано усі поставлені задачі.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Recommender system [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/Recommender_system>
2. Collaborative filtering [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/Collaborative_filtering>
3. Колаборативна фільтрація [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0_%D1%84%D1%96%D0%BB%D1%8C%D1%82%D1%80%D0%B0%D1%86%D1%96%D1%8F>
4. MatrixFactorization [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.trainers.matrixfactorizationtrainer?view=ml-dotnet-preview>
5. Метод стохастичного градієнта [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D1%81%D1%82%D0%BE%D1%85%D0%B0%D1%81%D1%82%D0%B8%D1%87%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B3%D1%80%D0%B0%D0%B4%D1%96%D1%94%D0%BD%D1%82%D0%B0#:~:text=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%20%D1%81%D1%82%D0%BE%D1%85%D0%B0%D1%81%D1%82%D0%B8%D1%87%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%B3%D1%80%D0%B0%D0%B4%D1%96%D1%94%D0%BD%D1%82%D0%B0%5B%D1%80%D0%B5%D0%B4.&text=stochastic%20gradient%20descent%2C%20incremental%20gradient,%D1%81%D0%BF%D1%83%D1%81%D0%BA%D1%83%20%D0%B7%D0%B0%20%D0%B4%D0%BE%D0%BF%D0%BE%D0%BC%D0%BE%D0%B3%D0%BE%D1%8E%20%D1%81%D1%82%D0%BE%D1%85%D0%B0%D1%81%D1%82%D0%B8%D1%87%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%BD%D0%B0%D0%B1%D0%BB%D0%B8%D0%B6%D0%B5%D0%BD%D0%BD%D1%8F.>
6. Tutorial: Build a movie recommender using matrix factorization with ML.NET [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.microsoft.com/en-us/dotnet/machine-learning/tutorials/movie-recommendation>

**ДОДАТКИ**

**Додаток А. Код LearningService**

public class LearningService: BaseService, ILearningService {

public LearningService (TripDbContext dbContext, IMapper mapper): base (dbContext, mapper) {}

public void Learn () {

var mlContext = new MLContext ();

var trainingDataView = LoadData(mlContext);

ITransformer model = BuildAndTrainModel (mlContext, trainingDataView);

SaveModel (mlContext, trainingDataView.Schema, model);}

private List<TripRating> GetAllRating () {

var rating = \_dbContext.Rating.ToList();

var preparedRating = \_mapper.Map<List<TripRating>>(rating);

return preparedRating;}

private IDataView LoadData (MLContext mlContext) {

var rating = GetAllRating ();

var trainingDataView = mlContext.Data.LoadFromEnumerable(rating);

return trainingDataView;}

private static ITransformer BuildAndTrainModel (MLContext mlContext, IDataView trainingDataView) {

IEstimator<ITransformer> estimator = mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "UserIdEncoded", inputColumnName: "UserId")

.Append(mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "TripIdEncoded", inputColumnName: "TripId"));

var options = new MatrixFactorizationTrainer.Options {

MatrixColumnIndexColumnName = "UserIdEncoded",

MatrixRowIndexColumnName = "TripIdEncoded",

LabelColumnName = "Label",

NumberOfIterations = 1000,

ApproximationRank = 100};

var trainerEstimator = estimator.Append(mlContext.Recommendation().Trainers.MatrixFactorization(options));

ITransformer model = trainerEstimator.Fit(trainingDataView);

return model;}

private static void SaveModel (MLContext mlContext, DataViewSchema trainingDataViewSchema, ITransformer model) {

mlContext.Model.Save(model, trainingDataViewSchema, ModelPath); }}

**Додаток Б. Код PredictingService**

public class PredictingService: BaseService, IPredictingService {

private readonly PredictionEnginePool<TripRating, TripRatingPrediction> \_model;

public PredictingService (PredictionEnginePool<TripRating, TripRatingPrediction> model,

TripDbContext dbContext, IMapper mapper): base (dbContext, mapper) {

\_model = model;}

public List<TripWithRatingModel> GetRecommendedForUser (int userId) {

var allTripsIds = \_dbContext.Rating.Select(t => t.TripId).Distinct();

var ratedTrips = \_dbContext.Rating.Where(r => r.UserId == userId).Select(t => t.TripId);

if (!ratedTrips.Any())

{

return new List<TripWithRatingModel> ();

}

var unratedTrips = allTripsIds.Except(ratedTrips).Join(\_dbContext.Trips, r => r, t => t.Id, (r, t) => t).ToList();

var mappedTrips = \_mapper.Map<List<TripWithRatingModel>>(unratedTrips);

foreach (var trip in mappedTrips)

{

var prediction = PredictRating (new TripRating {UserId = userId, TripId = trip.Id });

trip.Rating = prediction.Score;

}

return mappedTrips;

}

public TripRatingPrediction PredictRating (TripRating tripRating)

{

var prediction = \_model.Predict(tripRating);

return prediction;

}

}